CS 336 Lecture 28– 4/27/98

Announcements

- Program listings and some output due today.
- Quick report summarizing your data and comparing it to Feldmeier's due Friday.

Reliability at the Transport Layer

1. A major function of many transport layer protocols is to ensure reliable delivery. This is certainly true of the Internet's TCP.

   This is a bit odd for several reason:

   - You may have thought we handled the reliability issue a long time ago in the data link layer. Why do we have to deal with it again.
   - Even after you accept that we have to deal with it again, you might worder why we don't take care of it at the network layer (i.e. in IP).

   The following discussion will attempt to clear up these issues.

2. First, you must realize that even if low level protocols are designed to be reliable, network layer protocols may have to do additional work to preserve this reliability.

   - If an intermediate router crashed after receiving and acknowledging a message but before forwarding it, the message might still be lost

3. In the case of a network layer protocol like IP, this isn't even an issue.

   - IP, of course, is designed to function on a wide range of underlying networks. So, it is best to assume some of them might not provide reliability guarantees.

   So, we have to ask is it worth the effort to make our network layer protocol reliable.

4. By design, IP does not provide any guarantees of reliable delivery. There are several reasons for this.

   - While it might seem obvious, there are network applications where reliability is not a desirable thing if the cost is a loss in performance somewhere else.
     - The canonical example of this is speech where a few lost packets are less critical than many delayed packets.

     Leaving IP simple preserves the choice such applications need to sacrifice reliability for performance.
   - IP is still one step away from an end-to-end connection. If we really want confirmation of a message's delivery, we will want to know that it got to the target process not just the target machine. (In fact, we might even want to know that it was processed by rather than just delivered to the target process).

5. UDP, like IP, does not guarantee delivery. Even the UDP checksum is optional. Thus, UDP is intended for end-to-end applications where reliability is not a primary concern.

6. Within the Internet Protocol Suite, however, there is a reliable transport layer named TCP (Transmission Control Protocol).

7. TCP shares many of UDP's features.

   - It is implemented by encapsulating TCP packets (actually called segments) in IP packets
   - It depends on port numbers (that are interpreted independently from UDP port numbers).

8. TCP, however, differs from UDP in many ways.

   - It guarantees reliable, ordered delivery of data.
   - It is byte-stream oriented. That is, it does not offer to deliver packets but bytes. For example, even though a sending process invokes TCP 5 time sending 120 bytes of data each time, the receiving TCP may deliver 3 units of 200 bytes each to the process at the other end.

     Basically, the bytes sent will be delivered in the order sent but not necessarily in the same units.

9. One can learn a bit more about how TCP works by looking briefly at the fields in a TCP packet:

<table>
<tr><td colspan="3">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="4">Sequence Number</td></tr>
<tr><td colspan="4">Acknowledgement Number</td></tr>
<tr><td>Hdr. Len.</td><td>0</td><td>Flags</td><td>Window Size</td></tr>
<tr><td colspan="3">Checksum</td><td>Urgent Ptr.</td></tr>
<tr><td colspan="4">Options</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

10. If you look at the TCP packet fields, there is an oddity that suggests a big difference between ARQ at the data link and transport layers.

    - The "window" field specifies how much buffer space is available on the sender's side. As such it places a limit on the amount the other side can send before receiving acknowledgments (which may also contain adjustments of the window size).
    - The odd thing is that the window size field is much smaller than the sequence number field.
    - In our earlier discussion of data link protocols, the range of sequence numbers only had to be twice as big as the size of the buffers used. In TCP sequence numbers are 2^{16} bigger than the largest window size.
    - This is because datalink ARQ protocols were based on an assumption that is no longer true. At the data link layer, packets were assumed to be delivered in order (or not at all). At or above the network layer we have to accept the possibility that packets may be reordered.
    - To make sequence numbers work without the ordered delivery assumption, we must depend on the weaker assumption that packet lifetimes are bounded and use a range of sequence numbers that will not wrap around within one packet's lifetime.

11. In the data link level ARQ protocols, we never really addressed the issue of how the two ends of a connection got synchronized in the first place. This issue is harder to ignore now.